Projekt nr.1

Poniższy projekt ma na celu zaimplementowanie oraz porównanie algorytmu Quicksort z partycjonowaniem na pięć sposobów.

Część 1- implemantacja algorytmów

1. Lomuto

```
def partition_lomuto(arr,left,right):
    j=left
    pivot=arr[right]
    for i in range(left,right):
        if arr[i]<pivot:
            arr[j],arr[i]=arr[i],arr[j]
            j+=1

    arr[j],arr[right]=arr[right],arr[j]
    return j

def quicksort_lomuto(arr,left,right):
    if left>=right:
        return
    pivot=partition_lomuto(arr,left,right)
    quicksort_lomuto(arr,left,pivot-1)
    quicksort_lomuto(arr,pivot+1,right)
```

2. Lomuto z losowniem pivota jako z mediany trzech elementów

```
def partition_lomuto_med(arr,left,right):
    mid=(left+right)//2
    if arr[mid]<arr[left]:
        arr[left],arr[mid]=arr[mid],arr[left]
    if arr[right]<arr[left]:
        arr[right],arr[left]=arr[left],arr[right]
    if arr[mid]<arr[right]:
        arr[mid],arr[right]=arr[right],arr[mid]
    pivot=arr[right]
    j=left
    for i in range(left,right):
        if arr[i]<pivot:
            arr[j],arr[i]=arr[i],arr[j]
            j+=1

    arr[j],arr[right]=arr[right],arr[j]
    return j

def quicksort_lomuto_med(arr,left,right):
    if left >=right:
        return
    pivot=partition_lomuto_med(arr,left,right)
    quicksort_lomuto_med(arr,left,pivot-1)
    quicksort_lomuto_med(arr,pivot+1,right)
```

3. Hoare

```
def partition_hoare(arr,left,right):
    pivot=arr[left]
    i=left-1
    j=right+1
    while True:
        i+=1
        while arr[i]<pivot:
            i+=1
        j-=1
        while arr[j]>pivot:
            j-=1
        if i>=j:
            return j
        arr[i],arr[j]=arr[j],arr[i]

def quicksort_hoare(arr,left,right):
    if left>=0 and right>=0 and left<right:
        pivot=partition_hoare(arr,left,right)
        quicksort_hoare(arr,left,pivot)
        quicksort_hoare(arr,pivot+1,right)
    return
```

4. Dutch flag

```
def partition_dutch(arr,left,right):
    pivot=arr[(left+right)//2]
    lt=left
    eq=left
    gt=right
    while eq<=gt:
        if arr[eq]<pivot:
            arr[eq],arr[lt]=arr[lt],arr[eq]
            lt+=1
            eq+=1
        elif arr[eq]>pivot:
            arr[eq],arr[gt]=arr[gt],arr[eq]
            gt-=1
        else:
            eq+=1
    return lt,gt

def quicksort_dutch(arr,left,right):
    if left<right:
        lt,gt=partition_dutch(arr,left,right)
        quicksort_dutch(arr,left,lt-1)
        quicksort_dutch(arr,gt+1,right)
```

5. Dutch flag z losowaniem piovota jako mediany z trzech elementów

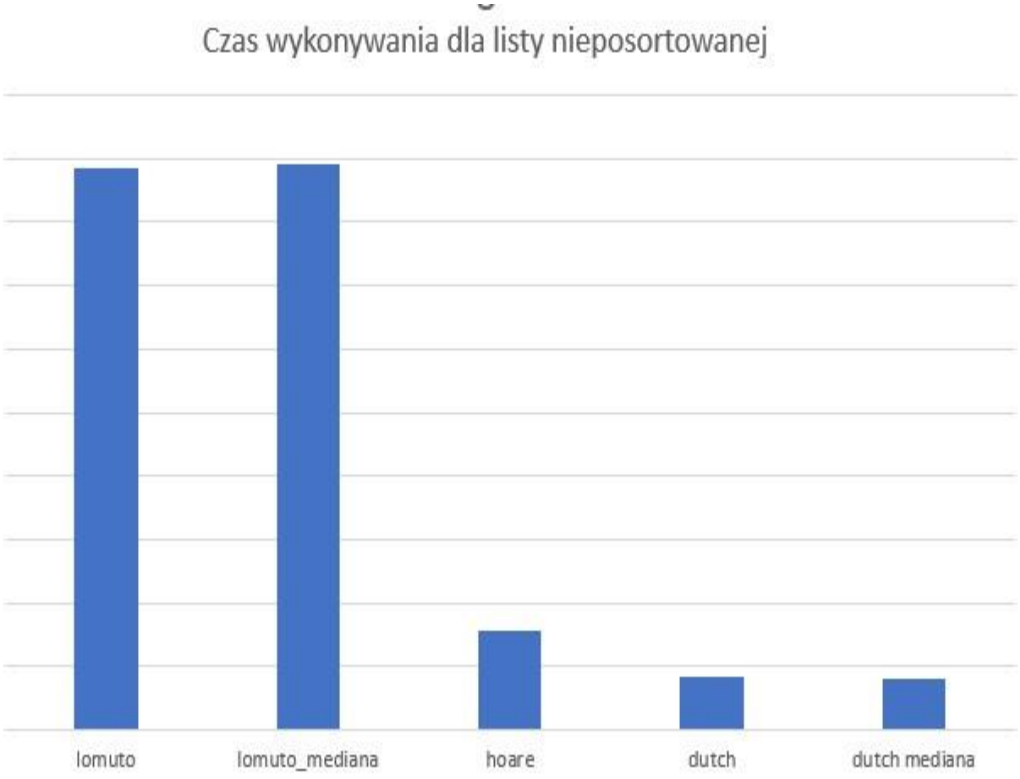
```
def partition_dutch_med(arr,left,right):
    median_elements=[arr[left],arr[(left+right)//2],arr[right]]
    pivot=statistics.median(median_elements)
    lt=left
    eq=left
    gt=right
    while eq<=gt:
        if arr[eq]<pivot:
            arr[eq],arr[lt]=arr[lt],arr[eq]
            lt+=1
            eq+=1
        elif arr[eq]>pivot:
            arr[eq],arr[gt]=arr[gt],arr[eq]
            gt-=1
        else:
            eq+=1
    return lt,gt


def quicksort_dutch_med(arr,left,right):
    if left<right:
        lt,gt=partition_dutch_med(arr,left,right)
        quicksort_dutch_med(arr,left,lt-1)
        quicksort_dutch_med(arr,gt+1,right)
```

Część 2- porównanie czasu działania algorytmów

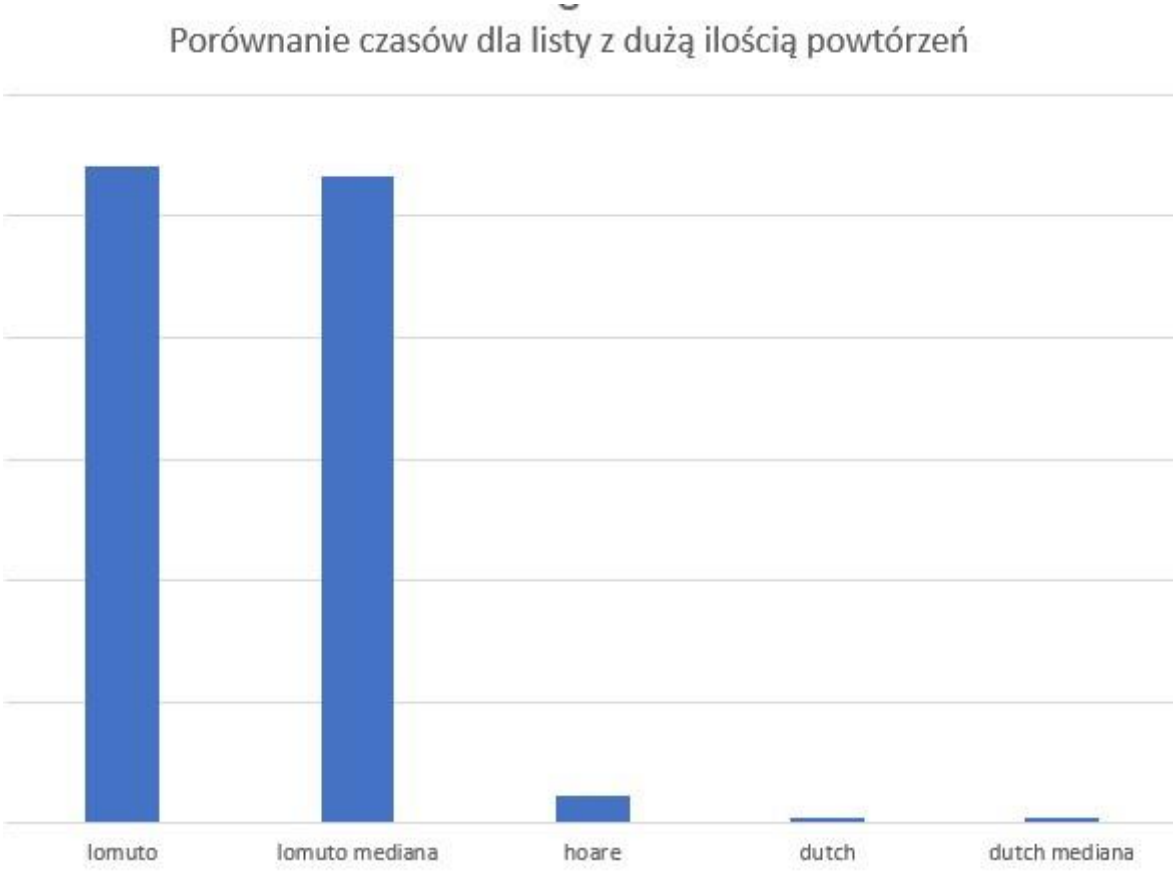
Będę tutaj porównywał czasy wykonywania powyższych programów dla trzech przypadków. Dla listy nieposortowanej zawierającej wartości z przedziału (1-100), dla listy nieposortowanej zawierającej dużo powtórzeń (elementy listy ta przybierają wartości z zakresu 1-5) oraz dla listy już posortowanej. Ilości elementów tych list wynoszą odpowiednio 50000, 4000, 2000. Podane czasy są średnią z 50 powtórzeń każdego programu.

- Dla listy nieposortowanej

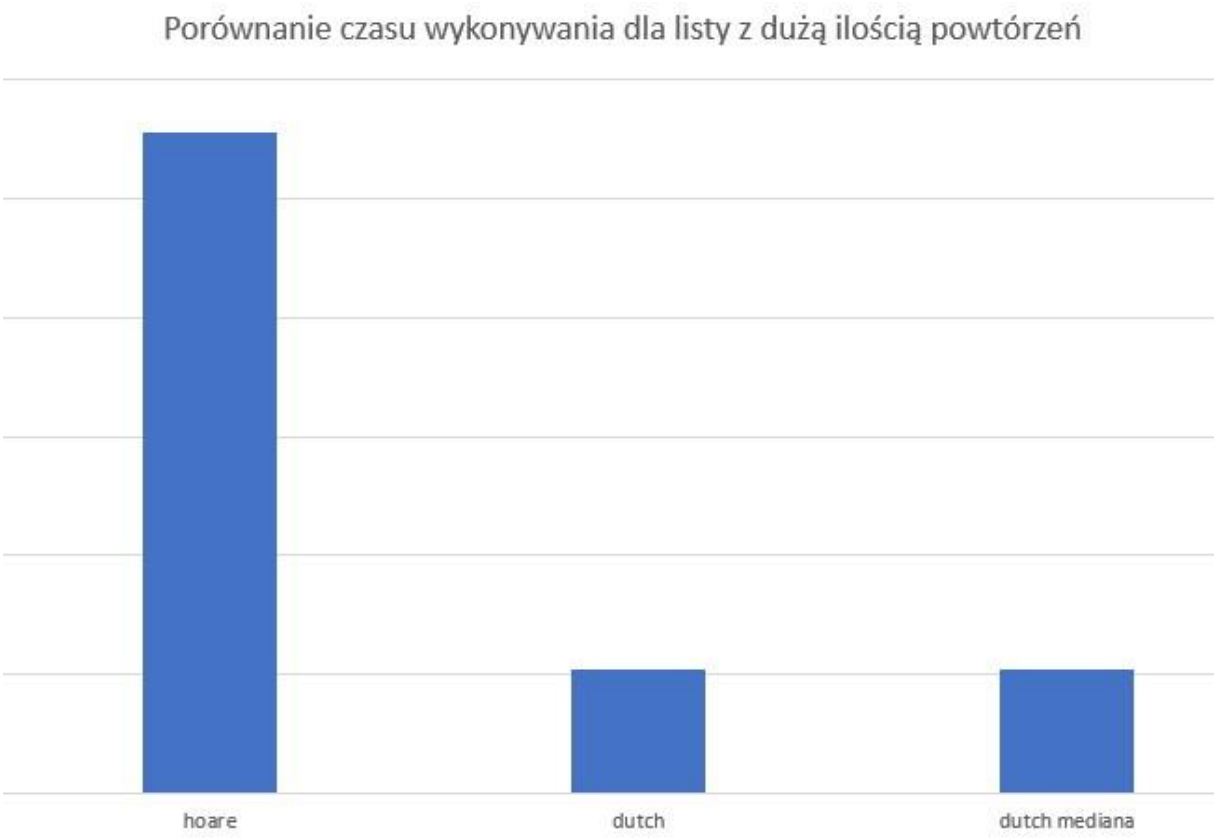


Jak widać najwięcej czasu zajmuje wykonanie algorytmów lomuto w obu wersjach, hoare radzi sobie całkiem nieźle, a najszybciej wykonują się algorytmy w wersji dutch flag- oba z podobnymi czasami

- Dla listy z dużą ilością powtórzeń

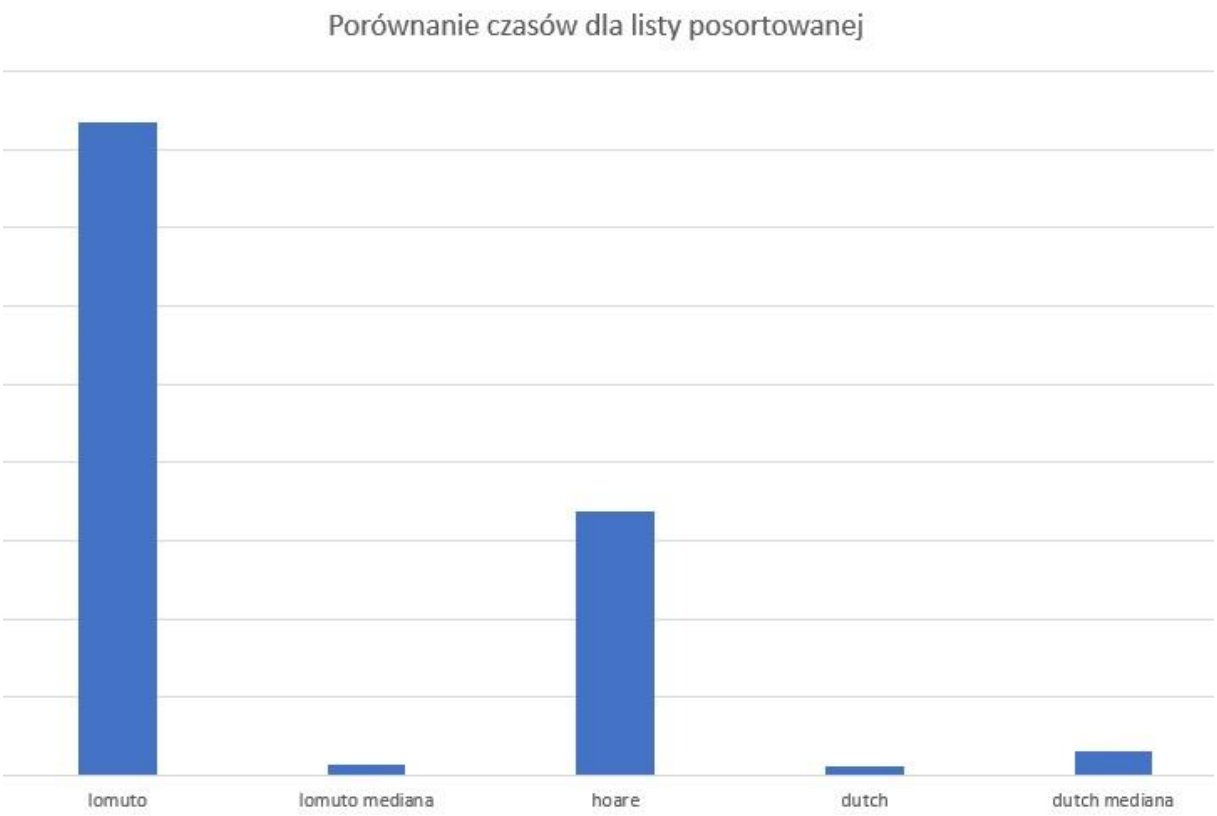


Tutaj również lomuto wykonuje się najwolniej, aby dokładniej zobaczyć który algorytm jest najwydajniejszy przedstawię wykres pozostałych wersji

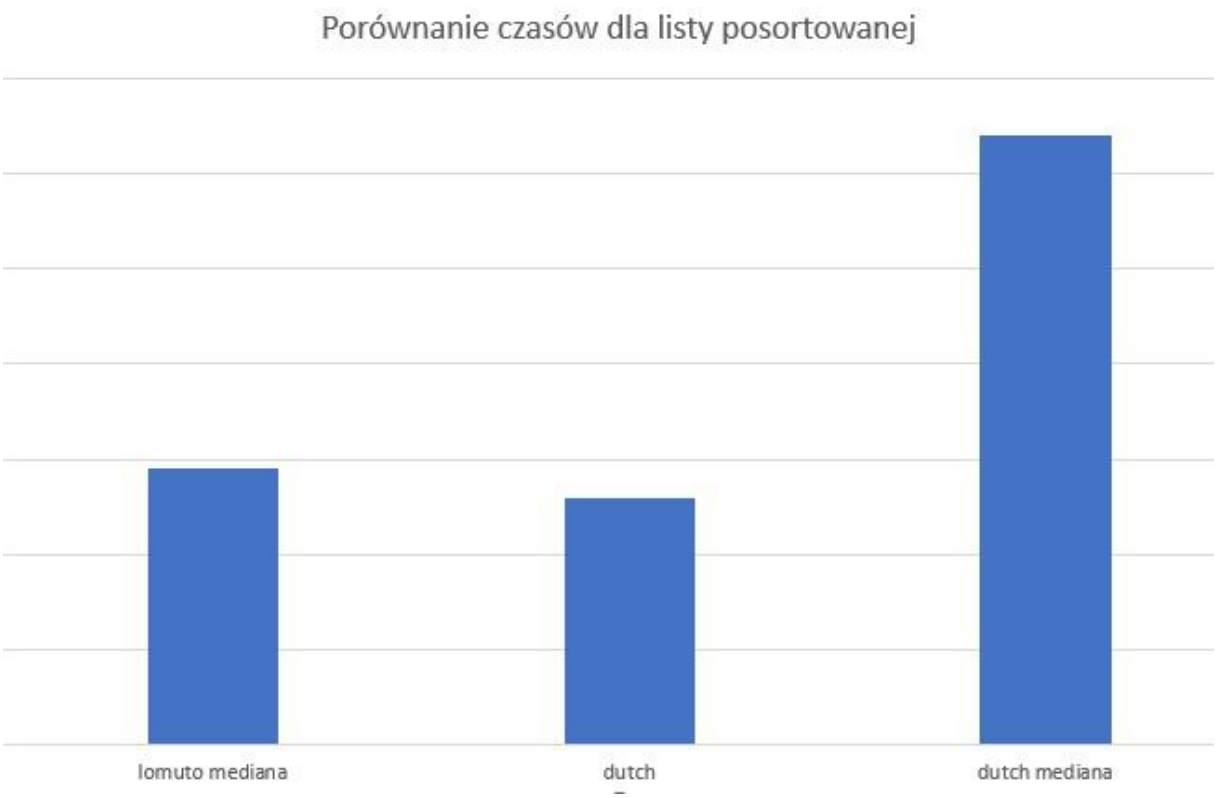


Widać tutaj wyraźną przewagę algorytmów typu Dutch flag, które wykonują się zdecydowanie najszybciej

- Dla listy posortowanej



W tym przypadku widać, że algorytmy horae oraz lomuto zostają w tyle, tutaj również przedstawię osobny wykres dla pozostałych wersji



Dutch z medianą wykonuje się najwolniej z tej trójki, natomiast lomuto z medianą oraz klasyczny Dutch flag wykonują się w bardzo podobnym czasie.

Po przeanalizowaniu wykresów okazuje się że najszybsze są wykresy typu Dutch flag